



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/600,932	06/20/2003	Anisoara Nica	SYB/0089.01	6192
31779	7590	01/30/2006	EXAMINER	
JOHN A. SMART 708 BLOSSOM HILL RD., #201 LOS GATOS, CA 95032-3503				HICKS, MICHAEL J
ART UNIT		PAPER NUMBER		
		2165		

DATE MAILED: 01/30/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/600,932	NICA, ANISOARA
	Examiner	Art Unit
	Michael J. Hicks	2165

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) Responsive to communication(s) filed on 6/20/2003.
 2a) This action is FINAL. 2b) This action is non-final.
 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) Claim(s) 1-49 is/are pending in the application.
 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
 5) Claim(s) _____ is/are allowed.
 6) Claim(s) 1-49 is/are rejected.
 7) Claim(s) _____ is/are objected to.
 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) The specification is objected to by the Examiner.
 10) The drawing(s) filed on 20 June 2003 is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____
3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date <u>12/29/2003</u>	5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)
	6) <input type="checkbox"/> Other: _____

DETAILED ACTION

1. Claims 1-49 pending

Double Patenting

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the “right to exclude” granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1-7, 10, 16-17, 22-25, 36-37, 40-43, 45, and 49 provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over Claims 1-22 of copending Application No. 10/835,230. Although the conflicting claims are not identical, they are not patentably distinct from each other because although the Claims from the instant application are directed towards constructing an optimal execution plan for a query, and the Claims in Application 10/835,230 are

directed towards the optimization of a query, both use the same method of using a query optimization graph for query blocks which represent portions of the database query, generating an optimal access plan for each query block through the use of sub-queries to determine favorable access plans, and the use of costing and pruning.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claim 25 rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claim 25 is directed towards a downloadable set of computer-executable instructions, which is merely an arrangement of data and therefore considered non-statutory. The rejection can be overcome by indicating a storage medium such as "a downloadable set of computer-executable instructions stored on a server".

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claim 9 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 9 states that all sets of “outer references” must have an access plan defined in the optimization process. The Examiner asserts that, although the term “outer references” appears in the specification, it appears only within algorithms or in reference to its use in those algorithms and is never defined. For these reasons the term “outer references” is considered to be indefinite. For the purpose of further examination the Examiner will take the term “outer references” to mean tables which are involved in an outer join.

Claim 19 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 19 states that the property vector includes “pipeline characteristics”. The Examiner asserts that, although the term “pipeline characteristics” appears in the specification, it is never defined and used only in reference to such characteristics being included in the property vector. Because no pipeline is discussed in the specification, there is a question as to what subject matter Applicant intends to claim. For these reasons the term “pipeline characteristics” is considered to be indefinite. For the purpose of further examination the Examiner will take the term “pipeline characteristics” to mean the order in which the operations of the execution strategy will be executed, e.g. the order in which the operations will enter the pipeline.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 1-25, 36-40, and 42-49 rejected under 35 U.S.C. 102(b) as being anticipated by Bowman et al. ("Join Enumeration in a Memory-Constrained Environment", Proceedings, 16th IEEE Data Engineering Conference, San Diego, California; March 2000 and referred to hereinafter as Bowman).

As per Claims 1, 24, and 25, Bowman discloses in a database system, a method for constructing an optimal query execution plan for executing a query (i.e. "*In today's computing environment, database technology can be found on virtually any device...The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that a method for optimizing a query in a database system is presented.) (Abstract), the method comprising: receiving a query specifying at least one join condition between two or more database tables (i.e. "*The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that the query has a high join degree (e.g. specifies at least one join between two or more database tables).) (Abstract); identifying each query block within said query, each query block comprising an atomic portion of said query (i.e. "*The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1). A QOG is the internal representation of a complete SQL statement2, possibly composed of multiple*

subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG." The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query, which each subquery/query block representing an atomic portion of the query (e.g. tables, predicates, etc.).) (Page 2, Column 2, Paragraph 4); creating subplans for each query block based on grouping portions of each query block (i.e. "...*the plan generation phase optimizes each subquery in the QOG independently, starting with the leaves...enumerate join strategies and prune the search space using a branch and bound heuristic.*" The preceding text excerpt clearly indicates that subplans are created for each subquery/query block, and that the subplans are based on enumerating join strategies (e.g. ways of grouping portions of each query block.) (Page 3, Column 1, Paragraph 3; Column 2, Paragraph 1); determining at least one favorable access plan for each subplan of each query block, said at least one favorable access plan determined based at least in part on estimated execution costs (i.e. "Cost estimation is an integral part of the enumeration algorithm, because it is through comparing the costs of partial access plans that the ASA optimizer can quickly prune significant portions of the join strategy search space...determine the cheapest strategy and construct the detailed access plan for that strategy." The preceding text excerpt clearly indicates that at least the cheapest/on favorable access plan is recalled/determined for each subplan of each query block, where the access plan is determined to be favorable based on execution cost estimation.) (Page 3, Column 1, Paragraph 2; Column 2, Paragraph 1); generating an optimal access plan for each query block based upon said at least one favorable access plan determined for each subplan (i.e. "...*determine the cheapest strategy and construct the detailed access plan for that strategy.*" The preceding text excerpt clearly indicates that the cheapest/optimal access plan is found for each subquery/query block which is based on the favorable access plan determined for each subplan.) (Page 3, Column 2, Paragraph 1); and constructing an optimal query execution plan based upon said optimal access plan generated for each query block (i.e. "...*a subsequent set of strategies is retained only if each ones cost is not provably*

greater than the cost of the best join strategy discovered thus far." The preceding text excerpt clearly indicates that the final, optimal query plan is based upon the set of optimal strategies (e.g. access plans) which were generated for each subquery/query block.) (Page 4, Column 2, Paragraph 5; Page 5, Column1, Paragraph 1).

As per Claim 2, Bowman discloses said step of identifying each query block within said query includes building a query optimization graph for each query block (i.e.

"The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1). A QOG is the internal representation of a complete SQL statement2, possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG...the plan generation phase optimizes each subquery in the QOG independently..." The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query while building a Query Optimization Graph for the query, which include query optimization graphs for each subquery/query block.) (Page 2, Column 2, Paragraph 4; Page 3, Column 1, Paragraph 3).

As per Claim 3, Bowman discloses generating a query optimization graph for each query block (i.e. *"The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1). A QOG is the internal representation of a complete SQL statement2, possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG...the plan generation phase optimizes each subquery in the QOG independently..."* The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query while building a Query Optimization Graph for the query, which include query

optimization graphs for each subquery/query block.) (Page 2, Column 2, Paragraph 4; Page 3, Column 1, Paragraph 3).

As per Claim 4, Bowman discloses said step of generating a query optimization graph includes generating subplans for each query block (i.e. "...*the plan generation phase optimizes each subquery in the QOG independently...enumerate join strategies and prune search space using a branch-and-bound heuristic.*" The preceding text excerpt clearly indicates that join strategies/subplans are generated for each subquery/query block) (Page 3, Column 1, Paragraph 3; Page 3, Column 2, Paragraph 1).

As per Claim 5, Bowman discloses said step of generating a query optimization graph includes generating plan nodes for each subplan, said plan nodes for joining tables and subplans (i.e. "*The join graph constructed for each subquery in the QOG is a graph G = <V,E>. The vertices in V[G] are of two types: simple vertices V^T, each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands). The edges in E[G] are of four types...For a valid SQL query the subgraph G' = <V,D> must be acyclic.*" The preceding text excerpt along with Figure 2 clearly indicates that a join graph is constructed including plan nodes for each subplan. Note that the plan nodes are represented by tables and joins which are connected by vertices to represent the subplan.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 6, Bowman discloses said step of generating a query optimization graph includes generating an array of access methods for each plan node (i.e. "*Plan generation involves the generation of alternative join strategies for each subquery block.*" The preceding

text excerpt clearly indicates that alternative/an array of join strategies/access methods are generated for each subquery. Note that the evaluation of each subquery involved constructing the join graph, which includes the plan nodes to which the join strategies/access methods are applied.) (Page 3, Column 1, Paragraph 2).

As per Claim 7, Bowman discloses said step of generating a query optimization graph includes generating an array of join methods for each plan node (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands).*") The preceding text excerpt clearly indicates that alternative/an array of join methods are generated for each subquery (e.g. left outer join, right outer join, etc.). Note that the evaluation of each subquery involved constructing the join graph, which includes the plan nodes to which the join methods are applied.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 8, Bowman discloses a subplan represents a table expression of a query block (i.e. Example 1 and the corresponding Figure 2 clearly indicates that the subplans constructed for the subquery/query block may represent table expressions from that subquery/query block as Figure 1 illustrated joins of tables (e.g. table expressions) for the subqueries/query blocks in Example 1.) (Example 1, Table 2).

As per Claim 9, Bowman discloses said step of determining at least one favorable access plan for each subplan includes generating at least one access plan for each different set of outer references (i.e. "...*in the case of outer joins, the set of valid join*

strategies is restricted to those where each outer join preserved tables must precede its null-supplying tables..." The preceding text excerpt clearly indicates that outer joins strategies/access plans are considered/generated for each table for which an outer join would produce a valid result (e.g. the set of outer references).) (Page 4, Column 2, Paragraph 2).

As per Claim 10, Bowman discloses the step of determining at least one favorable access plan for each subplan includes the substeps of: placing a candidate plan segment in the next position in a current access plan being generated, said candidate plan segment representing a particular plan node, access method and join method valid at said next position (i.e. "*Join enumeration is a recursive algorithm which iteratively adds another table to the prefix of a join strategy, whose length is denoted L_P , until the strategy is completely determined, at which point the strategy's cost is estimated. There may be several alternative tables that could be placed at any 'position' $LP + 1$ in the join strategy—a necessary condition is that the valid alternatives are those that satisfy a topological sort of the vertices V^T with respect to edges in E^D and the current prefix, denoted (V^T, D, L_P) .*" The preceding text excerpt clearly indicates that plan nodes (e.g. tables and they're associated methods of joining and access methods) are placed in the join strategy/access method if the plan segment is valid. Note that procedure is covered more completely in the detailed algorithm section.) (Page 4, Column 2, Paragraph 3); evaluating the current access plan including said candidate plan segment (i.e. "*Once the valid alternatives are selected, the enumeration algorithm considered them in rank order. Each table is assigned a 'rank' as to the suitability of that table in that position...in the current strategy.*" The preceding text excerpt clearly indicates that the Join strategy/access plan including he candidate plan segment is evaluated by assigning the plan segment a rank.) (Page 4, Column 2, Paragraph 4); and if the current access plan is less favorable than a favorable access plan previously identified, replacing said candidate

plan segment with another available candidate plan segment and repeating said evaluating substep (i.e. "*Join enumeration is a recursive algorithm which iteratively adds another table...*" The preceding text excerpt clearly indicates that the process is repeated until the most favorable table and associated join information/plan segment is found.) (Page 4, Column 2, Paragraph 3).

As per Claims 11 and 39, Bowman discloses a plan node comprises a quantifier object (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands).*" The preceding text excerpt clearly indicates that the plan nodes in the join graph include quantifiers over a base or derived table.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 12, Bowman discloses said quantifier object represents a base table (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands).*" The preceding text excerpt clearly indicates that the plan nodes in the join graph include quantifiers over a base or derived table.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 13, Bowman discloses said quantifier object represents a derived table (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived*

table, and outer join vertices V^l which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands)." The preceding text excerpt clearly indicates that the plan nodes in the join graph include quantifiers over a base or derived table.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claims 14 and 38, Bowman discloses a plan node comprises a subplan object (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^l which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands).*" The preceding text excerpt clearly indicates that the plan nodes in the join graph comprise subplan objects (e.g. vertices which represent joins).) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 15, Bowman discloses said subplan object represents a join tree of a table expression of a query block (i.e. Example one and Figure 2 clearly indicates that the vertices which represent joins are used to construct a join tree for a table expression from a subquery/query block (e.g. Figure 2 represents a join tree formed for a subquery/query block from Example 1).) (Figure 2, Example 1).

As per Claim 16, Bowman discloses if the current access plan is more favorable than a favorable access plan previously identified, determining whether the current access plan comprises a complete plan (i.e. "*Join enumeration is a recursive algorithm which iteratively adds another table to the prefix of a join strategy, whose length is denoted L_P , until the strategy is completely determined, at which point the strategy's cost is estimated.*" The preceding text excerpt

clearly indicates that the process continues until the strategy/access plan is complete. Note that procedure is covered more completely in the detailed algorithm section.) (Page 4, Column 2, Paragraph 3); if the current access plan is determined not to comprise a complete plan, retaining the current access plan and repeating the above substeps for placing a candidate plan segment in the next position of the current access plan (i.e. "*Join enumeration is a recursive algorithm which iteratively adds another table to the prefix of a join strategy, whose length is denoted L_P , until the strategy is completely determined, at which point the strategy's cost is estimated.*" The preceding text excerpt clearly indicates that the process continues until the strategy/access plan is complete.) (Page 4, Column 2, Paragraph 3); and otherwise, if the current access plan is determined to be a complete plan, retaining the current access plan as a favorable access plan and repeating the above substeps to consider other available alternatives to the current access plan while alternatives are available (i.e. "*Join enumeration is a recursive algorithm which iteratively adds another table to the prefix of a join strategy, whose length is denoted L_P , until the strategy is completely determined, at which point the strategy's cost is estimated...Once the valid alternatives are selected, the enumeration algorithm considered them in rank order. Each table is assigned a 'rank' as to the suitability of that table in that position...in the current strategy.*" The preceding text excerpt clearly indicates that once a complete strategy/plan is determined, each strategy/plan is ranked and the steps are repeated to determine the highest ranked complete plan.) (Page 4, Column 2, Paragraphs 3-4).

As per Claims 17 and 45, Bowman discloses generating a property vector for said current access plan including said candidate plan segment (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76).*" The

preceding text excerpt clearly indicates that the current access plan, which may be partial or complete, is saved if its cost is found to be lower than the current access plans. The method of saving the plan (which would include the candidate plan segment) could easily be a vector (and indeed appears to be indicated as a vector in Algorithm 3), which would also act as the data structure used to compare the current and 'best' access plans.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claims 18 and 46, Bowman discloses said property vector includes estimated execution costs for said current access plan (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76).*" The preceding text excerpt clearly indicates that the property vector, as disclosed above, also saves/includes the execution cost.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claim 19, Bowman discloses said property vector includes pipeline characteristics for said current access plan (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76).*" The preceding text excerpt clearly indicates that the property vector as disclosed above also includes pipeline characteristics (e.g. the ordering of the operations in the strategy) as these would have to be saves/included in order to recall and perform the access plan.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claim 20, Bowman discloses said property vector includes order properties for said current access plan (i.e. "ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76)." The preceding text excerpt clearly indicates that the property vector as disclosed above also includes order properties (e.g. in what order the operations in the strategy/access plan are performed) as these would have to be saves/included in order to recall and perform the access plan.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claims 21 and 44, Bowman discloses said step of generating at least one favorable access plan for each subplan of each query block includes using a left-deep join enumeration strategy (i.e. "*As described earlier, ASA uses left-deep processing trees exclusively...*" The preceding text excerpt clearly indicates that the join enumeration, which the excerpt is describing, uses only left deep processing trees, therefore making it a left deep join enumeration strategy.) (Page 4, Column1, Paragraph 2).

As per Claims 22 and 43, Bowman discloses said step of generating at least one favorable access plan for each subplan of each query block includes starting with the innermost nested subplans (i.e. "*The set of subquery blocks within a QOG form a tree, with the outermost SELECT block at the root...the plan generation phase optimizes each subquery in the QOG independently, starting with the leaves.*" The preceding text excerpt clearly indicates that because the outermost subquery is the root, the innermost nested subplans would be the leaves, where the query optimizer starts.) (Page 2, Column 2, Paragraph 4; Page 3, Column 1, Paragraph 3).

As per Claim 23, Bowman discloses said step of generating at least one favorable access plan for each subplan includes evaluating execution costs of partial access plans and pruning partial access plans less favorable than previously generated complete access plans (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76). Otherwise, the algorithm considers each candidate vertex in rank order. A prior sort of the vertices (line 81) ensures syntax-independence...Pruning of the search takes place on line 97, where if a recursive call to ENUMERATE returns a position less than L^P we bypass this entire prefix and return to consider a prefix with length $L_P - 1$.*" The preceding text excerpt clearly indicates that each partial strategy/execution plan is evaluated to determine if its cost is more favorable than previously generated plans, and if the cost is less favorable, the partial access plan is pruned.)
(Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1).

As per Claim 36, Bowman discloses in a database system, a method for optimizing execution of a query (i.e. "*In today's computing environment, database technology can be found on virtually any device...The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that a method for optimizing a query in a database system is presented.)
(Abstract), the method comprising: receiving a query specifying selection of data from a plurality of database tables (i.e. "*The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that the query has a high join degree (e.g. specifies at least one

join between two or more/a plurality of database tables.) (Abstract); enumerating candidate plan segments for inclusion in an access plan for selecting data specified by the query, said candidate plan segments representing alternative strategies for joining relations and selecting data (i.e. "...*the plan generation phase optimizes each subquery in the QOG independently, starting with the leaves...enumerate join strategies and prune the search space using a branch and bound heuristic.*" The preceding text excerpt clearly indicates that subplans are created for each subquery/query block, and that the subplans are based on enumerating join strategies (e.g. ways of grouping portions of each query block.) (Page 3, Column 1, Paragraph 3; Column 2, Paragraph 1); for each query block comprising an atomic portion of said query (i.e. "*The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1). A QOG is the internal representation of a complete SQL statement2, possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG.*" The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query, which each subquery/query block representing an atomic portion of the query (e.g. tables, predicates, etc.).) (Page 2, Column 2, Paragraph 4), determining an optimal access plan by performing the substeps of (i.e. "*In today's computing environment, database technology can be found on virtually any device...The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that an optimal access plan for a query in a database system is found.) (Abstract): placing a candidate plan segment in a partial access plan being generated for said query block (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74).*" The preceding text excerpt clearly indicates that each candidate vertex/plan segment is placed in a partially

complete access plan.) (Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1); evaluating said partial access plan including said candidate plan segment (i.e.

"ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74)."

The preceding text excerpt clearly indicates that each partial strategy/access plan is evaluated to determine its cost.) (Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1); if said partial access plan is less favorable than a complete access plan previously

identified for said query block, pruning said candidate plan segment (i.e. *"ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76).*

Otherwise, the algorithm considers each candidate vertex in rank order. A prior sort of the vertices (line 81) ensures syntax-independence...Pruning of the search takes place on line 97, where if a recursive call to ENUMERATE returns a position less than L^P we bypass this entire prefix and return to consider a prefix with length $L_P - 1$." The preceding text excerpt clearly indicates that each partial strategy/execution plan is evaluated to determine if its cost is more favorable than previously generated plans, and if the cost is less favorable, the partial access plan is pruned.) (Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1); otherwise, adding an additional candidate plan segment to said partial access plan and repeating the above steps until a complete access plan for said query block is generated (i.e. *"Join enumeration is a recursive algorithm which iteratively adds another table to the prefix of a join strategy, whose length is denoted L_P , until the strategy is completely determined, at which point the strategy's cost is estimated."* The preceding text excerpt clearly indicates that the join enumeration/adding of additional segments will continue until a complete plan is generated.)

(Page 4, Column 2, Paragraph 3); retaining a complete access plan if it is more favorable than other complete access plans previously generated for the query block (i.e. "Once the valid alternatives are selected, the enumeration algorithm considered them in rank order. Each table is assigned a 'rank' as to the suitability of that table in that position...in the current strategy." The preceding text excerpt clearly indicates that once the plan is complete, the complete plans are ranked and the complete plan with the best rank is retained.) (Page 4, Column 2, Paragraph 4); and otherwise pruning a complete access plan which is less favorable than other complete access plans (i.e. "Once the valid alternatives are selected, the enumeration algorithm considered them in rank order. Each table is assigned a 'rank' as to the suitability of that table in that position...in the current strategy." The preceding text excerpt clearly indicates that if the complete plans rank does not indicate it is the most favorable, it is pruned.) (Page 4, Column 2, Paragraph 4); and generating a query execution plan based upon the optimal access plan determined for each query block (i.e. "...a subsequent set of strategies is retained only if each ones cost is not provably greater than the cost of the best join strategy discovered thus far." The preceding text excerpt clearly indicates that the final, optimal query plan is based upon the set of optimal strategies (e.g. access plans) which were generated for each subquery/query block.) (Page 4, Column 2, Paragraph 5; Page 5, Column1, Paragraph 1).

As per Claim 37, Bowman discloses each said candidate plan segment comprises a plan node, an access method and a join method (i.e. Figure 2 clearly indicates that each candidate plan segment (which is represented in the join graph) comprises a plan node, an access method, and a join method all as detailed above.) (Figure 2).

As per Claim 40, Bowman discloses said substep of evaluating said partial access plan includes comparing estimated execution costs of said partial access plan to

estimated execution costs of a complete access plan previously generated for said query block (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76). Otherwise, the algorithm considers each candidate vertex in rank order.*" The preceding text excerpt clearly indicates that the execution costs of the partial strategies/plans are evaluated and then compared to the execution cost of previously saved/generated execution plans.) (Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1).

As per Claim 42, Bowman discloses said step of generating an optimal access plan for each query block includes generating an optimal access plan for each subplan of each query block (i.e. "...*a subsequent set of strategies is retained only if each ones cost is not provably greater than the cost of the best join strategy discovered thus far.*" The preceding text excerpt clearly indicates that the final, optimal query access plan is based upon the set of optimal strategies (e.g. access plans) which were generated for each subquery/query block.) (Page 4, Column 2, Paragraph 5; Page 5, Column1, Paragraph 1).

As per Claim 47, Bowman discloses said evaluating substep includes comparing said property vector of said partial access plan to a property vector of a complete access plan previously generated for said query block (i.e. "*ENUMERATE (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76).*" The preceding text

excerpt clearly indicates that the access plan (the ordering of which appears to be stored in a vector as disclosed above), which may be partial, is compared to the current best access plan, which may be complete. Note that the current best access plan will have its property vector saved for recall.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claim 48, discloses retaining a property vector of a complete plan generated for a query block if said complete plan is more favorable than any other complete access plan previously generated for the query block (i.e. "*ENUMERATE* (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76)." The preceding text excerpt clearly indicates that if the current strategy/access plan is found to be better than the currently saved best access plan, the property vector (as disclosed above) of the current aces plan is saved for recall.) (Algorithm 3; Page 5, Column 2, Paragraph 3).

As per Claim 49, Bowman discloses said substep of pruning said candidate plan segment includes replacing said candidate plan segment with another available candidate plan segment and repeating said evaluating substep (i.e. "*ENUMERATE* (Algorithm 3) is a recursive procedure that iteratively adds a vertex to the current join strategy pre-fix until only a single table remains, at which point the algorithm performs index selection and predicate placement (lines 70 through 73) and subsequently estimates the cost of the strategy (line 74). If that cost is lower than the best achieved thus far, the cost and strategy are saved for later recall (line 76). Otherwise, the algorithm considers each candidate vertex in rank order. A prior sort of the vertices (line 81) ensures syntax-independence...Pruning of the search takes place on line 97, where if a recursive call

to ENUMERATE returns a position less than L^P we bypass this entire prefix and return to consider a prefix with length $L_P - 1$." The preceding text excerpt clearly indicates that the algorithm is iterative and recursive, so if the outcome of a iteration results in a candidate segment being pruned, the algorithm will select a new candidate to replace the pruned segment (e.g. recursion) and start over and will continue to do so until there are no more iterations.) (Algorithm 3; Page 5, Column 2, Paragraphs 4-5; Page 6, Column1, Paragraph 1).

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

11. Claims 26-35 and 41 rejected under 35 U.S.C. 103(a) as being unpatentable over Bowman in view of Du et al. (U.S. Patent Number 5,649,591 and referred to hereinafter as Du).

As per Claim 26, Bowman discloses in a database system, a method for generating a bushy trees during optimization of a database query (i.e. "*A method for optimizing data retrieval from a multidatabase system by restructuring a database query tree to optimize query response time in a two step optimization process... Then this data is utilized in the balancing of the left deep join query tree so that the const for access to each left child subtree is substantially equal to the const for access to each right child subtree.*" The preceding text excerpt clearly indicates that a method

for generating a bushy (e.g. balanced) query tree during query optimization in a database system is presented.) (Abstract), the method comprising: receiving a database query specifying at least one join condition between two or more database tables (i.e. "*The algorithm is able to efficiently optimize complex queries with high join degree by employing a novel approach to cost based pruning of the search space.*" The preceding text excerpt clearly indicates that the query has a high join degree (e.g. specifies at least one join between two or more database tables).) (Abstract); identifying each query block within said query, said query block comprising an atomic block of said query (i.e. "*The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1).* A QOG is the internal representation of a complete SQL statement², possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG." The preceding text excerpt clearly indicates that subquery blocks are identified within the query, which each subquery block representing an atomic portion of the query (e.g. tables, predicates, etc.) (Page 2, Column 2, Paragraph 4); building a query optimization graph for each query block, said query optimization graph including plan nodes representing subplans and quantifiers of each query block (i.e. "*The input to the plan generation phase of Adaptive Server Anywhere's query optimizer is a Query Optimization Graph (QOG), pronounced 'cog' (see Figure 1).* A QOG is the internal representation of a complete SQL statement², possibly composed of multiple subquery blocks. The database entities referred to by each subquery block, including tables, columns, predicates, and so on, are also included in the QOG...Plan generation involves the generation of alternative join strategies for each subquery block...the plan generation phase optimizes each subquery in the QOG independently..." The preceding text excerpt clearly indicates that subquery/query blocks are identified within the query while building a Query Optimization Graph for the query, which include query optimization graphs for each subquery/query block and joins strategies/subplans and quantifiers for each block.) (Page 2, Column 2, Paragraph 4; Page 3, Column 1, Paragraphs 2-3); constructing a join tree

for each subplan based upon selecting access methods, join methods, and join order for plan nodes of said query optimization graph having favorable execution costs (i.e.

"...construct a join graph for the query that models inner and outer equijoin predicates...enumerate join strategies and prune the search space using a branch-and-bound heuristic...recall the cheapest strategy and construct the detailed access plan for that strategy." The preceding text excerpt clearly indicates that a join tree/graph is constructed using access methods/join strategies, join methods (e.g. inner and outer joins and left and right joins as stated above), join order for plan nodes (e.g. considered in the join enumeration), and the cheapest access plan is chosen (e.g. the execution cost is considered).) (Page 3, Column 1, Paragraph 4; Page 3, Column 2, Paragraph 1); constructing an optimal access plan for each query block based upon said join tree constructed for each subplan (i.e. "*The entire algorithm is branch-and-bound in the sense that a subsequent set of strategies is retained only if each one's cost is not provably greater than the cost of the best joins strategy discovered thus far.*") The preceding text excerpt clearly indicates that an optimal strategy/access plan is generated for each subplan which the join enumerator considers.) (Page 4, Column 2, Paragraph 5).

Bowman fails to disclose generating a bushy execution tree based upon the optimal access plan determined for each query block.

Du discloses generating a bushy execution tree based upon the optimal access plan determined for each query block (i.e. "*Then, this data is utilized in the balancing of the left deep join query tree so that the cost for access to each left child subtree is substantially equal to the cost for access for the right child subtree.*") The preceding text excerpt along with figures 12A-12F clearly indicates that the optimized access plan is transformed into a balanced/bushy execution tree.) (Figures 12A-12F; Abstract).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Bowman to include the teachings of Du to include

generating a bushy execution tree based upon the optimal access plan determined for each query block with the motivation to optimize data retrieval from a multidatabase system by restructuring a database query tree to optimize query response time (Du, Abstract).

As per Claim 27, Bowman discloses a query block comprises a selected one of a main block of a Structured Query Language (SQL) statement, a main block of a derived table, a main block of a view, a main block of a subquery used in a SQL statement, a derived table, and a view (i.e. "*Each subquery block – which, for example, can represent an input to a UNION operation, or a true subquery contained within an EXISTS predicate...*" The preceding text excerpt clearly indicates that a subquery/query block may be a main block of an SQL statement (as SQL is used as the example query language).) (Page 2, Column 2, Paragraph 4).

As per Claim 28, Bowman discloses said step of building said query optimization graph includes modeling left outer joins, right outer joins, and full outer joins as subplans which correspond to null-supplying sides of an outer join (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands). The edges in $E[G]$ are of four types. The first set of directed edges, termed an outer join edge and denoted E^J , relate each null-supplying table of a left-outer join to its (immediate) left-outer join vertex v (included in) V^J .*" The preceding text excerpt and Figure 2 clearly indicates that left and right outer joins are modeled in the subplans corresponding to null supplying sides of an outer join. Not

that while footnote 3 points out that full outer joins are not considered, it does not bar them from being used with the method.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 29, Bowman discloses said step of generating a query optimization graph includes generating an array of access methods for each plan node (i.e. "*Plan generation involves the generation of alternative join strategies for each subquery block.*" The preceding text excerpt clearly indicates that alternative/an array of join strategies/access methods are generated for each subquery. Note that the evaluation of each subquery involved constructing the join graph, which includes the plan nodes to which the join strategies/access methods are applied.) (Page 3, Column 1, Paragraph 2).

As per Claims 30, Bowman discloses said step of generating a query optimization graph includes generating an array of join methods for each plan node (i.e. "*The join graph constructed for each subquery in the QOG is a graph $G = \langle V, E \rangle$. The vertices in $V[G]$ are of two types: simple vertices V^T , each of which represents a quantifier over a base or derived table, and outer join vertices V^J which represent a left-outer join (all right-outer joins are rewritten as left-outer joins by swapping their operands).*" The preceding text excerpt clearly indicates that alternative/an array of join methods are generated for each subquery (e.g. left outer join, right outer join, etc.). Note that the evaluation of each subquery involved constructing the join graph, which includes the plan nodes to which the join methods are applied.) (Figure 2; Page 3, Column 2, Paragraph 2).

As per Claim 31, Bowman fails to disclose said step of constructing bushy join trees for each subplan includes using a left-deep enumeration strategy.

Du discloses said step of constructing bushy join trees for each subplan includes using a left-deep enumeration strategy (i.e. "*first a traditional cost-based (e.g. System R style) optimization process is employed to obtain a left-deep join execution plan which is optimal with respect to total resource usage; then the plan is improved (with respect to response time) using the proposed transformations.*" The preceding text excerpt clearly indicates that the bushy trees are constructed out of left deep trees which were optimized using an enumeration strategy.) (Column 4, Lines 37-41).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Bowman to include the teachings of Du to include said step of constructing bushy join trees for each subplan includes using a left-deep enumeration strategy with the motivation to optimize data retrieval from a multidatabase system by restructuring a database query tree to optimize query response time (Du, Abstract).

As per Claim 32, Bowman discloses said step of constructing a join tree includes evaluating execution costs of a candidate plan segment to be added to said join tree being constructed (i.e. "...recall the cheapest strategy and construct the detailed access plan for that strategy." The preceding text excerpt clearly indicates that the join graph/tree evaluates cost (e.g. to be able to determine the cheapest strategy) while it is being built.) (Page 3, Column 2, Paragraph 1).

As per Claim 33, Bowman discloses said candidate plan segment comprises a selected plan node together with an access method and a join method (i.e. Figure 2 clearly indicates that each candidate plan segment (which is represented in the join graph) comprises a plan node, an access method, and a join method all as detailed above.) (Figure 2).

As per Claim 34, Bowman fails to disclose said bushy tree comprises a processing tree having composite relations for left and right children of join nodes.

Du discloses said bushy tree comprises a processing tree having composite relations for left and right children of join nodes (i.e. "*Another object of the invention is to assume that the final execution plan is globally optimal with respect to total resource usage. This is the case since the second phase starts with a total cost optimal execution plan whereby each transformation will improve response time without an increase of total resource usage.*" The preceding text excerpt along with figure 9A-9F clearly indicates that because the plan the second phase (e.g. balancing phase starts with is optimal, it can be considered to have composite relations. Figures 9A-9F illustrate that because the balancing begins at the bottommost node of the left deep tree, the relations will remain composite in relation to the root node.) (Figures 9A-9F; Column 5, Lines 1-6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Bowman to include the teachings of Du to include said bushy tree comprises a processing tree having composite relations for left and right children of join nodes with the motivation to optimize data retrieval from a multidatabase system by restructuring a database query tree to optimize query response time (Du, Abstract).

As per Claims 35 and 41, Bowman fails to disclose said step of constructing an optimal bushy access plan includes evaluating execution costs of partial access plans enabling earlier pruning of unfavorable access plans.

Du discloses said step of constructing an optimal bushy access plan includes evaluating execution costs of partial access plans enabling earlier pruning of unfavorable access plans (i.e. "*A response time for the root query and for each of the query nodes is estimated and access response times to each table node and subtree are estimated...Pruning Non-Cost-Improving Transformations: Heuristics can be used to prevent transformations that are unlikely to be const improving, thereby reducing transformation overhead.*" The preceding text excerpt clearly indicates that execution costs for access plans (e.g. response times to table nodes and subtrees) are estimated which allows for early pruning of unfavorable access plans (e.g. access plans which do not provide an improvement.)) (Abstract; Column 18, Lines 40-43).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Bowman to include the teachings of Du to include said step of constructing an optimal bushy access plan includes evaluating execution costs of partial access plans enabling earlier pruning of unfavorable access plans with the motivation to optimize data retrieval from a multidatabase system by restructuring a database query tree to optimize query response time (Du, Abstract).

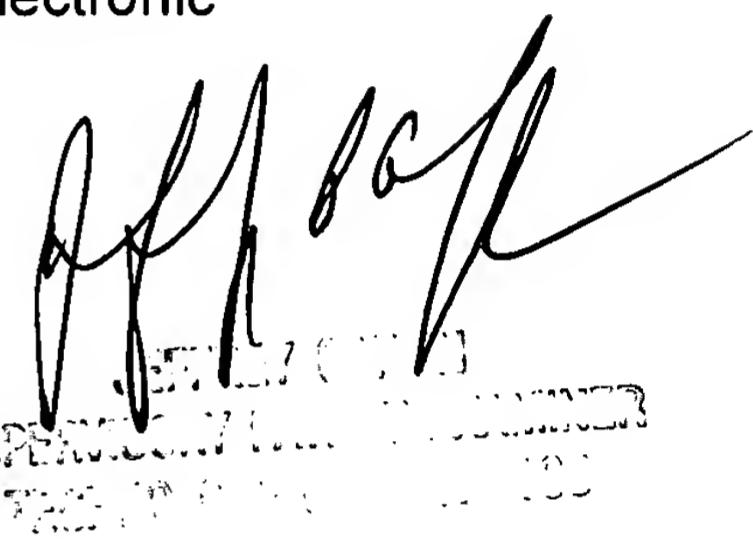
Points of Contact

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Friday 8:30a - 5:00p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Gaffin can be reached on (571) 272-4146. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Michael J Hicks
Art Unit 2165
(571) 272-2670


SUPERVISOR
7/22/2010